A Bandwidth Friendly Search Engine

Clare Bradford and Ian.W.Marshall
BT Labs, Martlesham Heath, Ipswich, UK. IP5 3RE.
e-mail :clare.bradford@bt.com and ian.w.marshall@bt.com

Abstract

The Internet plays host to many millions of documents and images and is increasing in size all the time. As a result locating web content is becoming increasingly difficult for users, and search traffic from users and spiders is increasing rapidly. A directory of the contents of the emerging cache hierarchy would be more complete than existing tools and avoid the need for spider traffic. However it is also essential to minimise the user traffic. A promising approach is to encourage users to refine their queries through categories. Automatic categorisation of a cache directory is demonstrated, and an adaptive categorisation scheme is proposed.

Introduction

The Internet has become a significant publication medium, it plays host to many millions of documents and images and is increasing in size all the time [1]. The massive size of the internet has created a market need for impressive navigational tools. The largest search engines currently cover less than half the cacheable contents of the Web [2], and are unable to accurately interpret the simple search terms preferred by users. Should search engine developers create bigger indexes to give better internet coverage, thereby increasing the numbers of spiders accessing and querying new sites ? We believe this would simply add to network congestion without benefit. It would merely increase the number of irrelevant search results the user is required to sort through, thereby increasing not only the spider traffic but the user traffic too. On the other hand, in order to provide better chances of finding material the user would find useful, the source database must clearly become more comprehensive.

The Harvest project partly addressed this by making directories from a cache [3]. Providing a query interface to a cache hierarchy will deliver two immediate benefits. It provides access to a directory which is more complete than any current search engine without requiring spiders to populate the database, and it enables the query responses to be ordered on the basis of file popularity, increasing the probability that useful pages are early in the response list.

The Harvest catalogue servers maintained a flat directory, similar to early search engines, that was not able to prioritise responses to match user requirements. This approach is not scalable since the user has to filter too many responses. We have attempted to address this issue by structuring the database and enabling the structure to adapt to changing user requirements.

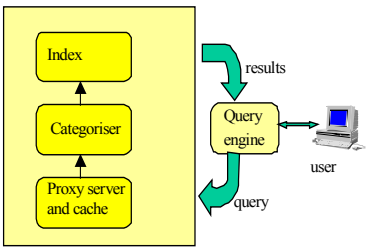
Ordering responses on the basis of file popularity is not sufficient – many popular pages are only relevant to a minority of queries but are written so as to give them a high probability of appearing in the response to any simple query. It is necessary to enable users to ask the search tool to filter responses which are not relevant, without significantly increasing query complexity. A common approach is to divide the directory into a number of categories which the user can select before entering the query.

In the next section we present a design for an adaptable categorised query engine based on a hierarchy of cache directories, and report some results derived from a partial implementation. Our initial implementation demonstrates for the first time that automated categorisation of a cache directory, with sufficient accuracy to be useful, is possible. Further work is required to implement adaptive categorisation.

Design

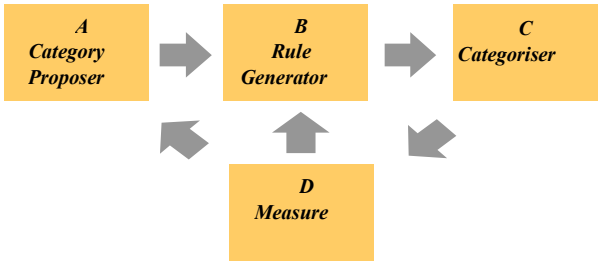
The design, shown in Figure 1, consists of a server based query engine that searches a local cache. The cache has been categorised and indexed and the query engine serves results from this sorted cache. The query engine, proxy server and indexer are adequately documented elsewhere [4], and we do not attempt to duplicate this here. The next part of this section concentrates on the categoriser.

Figure 1 System Design



A schematic of our design for an adaptive categoriser of a cache index is shown in Figure 2 and described in some detail below

Figure 2 Adaptive categorisation at a single node



The category Proposer (A) is a learning agent whose primary role is to collect and analyse information contained in the search index, and propose category lists to the rule generator (B). The analysis is initially based on wordcounts, metadata expressions, origin server, and update/time-to-live, but could be based on any information stored in the index. The analysis could be performed by a neural network, with adjustable weights for each input metric. The network would be trained on a standard dataset then allowed to adapt to the needs of its user community. The index itself uses a static schema but the proposer is intended to be adaptive, so that the agent will learn which metrics are most useful and be able to react to underlying trends which may render a particular metric more or less useful with time. It should also be able to respond to occasional new fields being added to the index, using an operator initiated retrain function. The proposer will have an internal feedback loop for learning purposes, enabling it to autonomously learn how best to propose a fixed number of categories with an evenly distributed population, and with comprehensible descriptors derived from metadata keywords or from popular query terms. External feedback to enable adjustment of metrics is provided by the Measure function (D). This feedback is primarily a list of popular query terms generated by users. A will only initiate a proposal when D supplies A with a significantly modified list of popular queries, or when D reports a significant failure such (e.g. falls in usage, repeated queries, and identical queries of multiple categories), or when an operator requests it. A sends the proposed categories, and a listing of its node weights to the rule generator (B).

The rule generator, B, is responsible for converting the neural net's weights into a set of rules that can be used by the categoriser, C. The rule generator is intended to use the techniques described in [5], governed by a set of metarules provided by the operator. The generator is able to modify its metarules in response to feedback from the measure function, from the operator, and from internal metrics. The internal metrics would include consistency checks and rulebase minimisation, but others may also be required.

The Categoriser, C, is the classification agent, which uses input from its own 'knowledge' and the rules given via B to associate objects in the index with categories. The same object can appear within many categories, but the categoriser will calculate a different degree of fit for each association, and store it in the index. This enables the query engine to order responses, to a search within a category, on the basis of fit to the category. It is envisaged that the user will have the option to specify different orderings of the response when they make a query. The object associations and metrics are located within a structured directory, resulting in fast retrieval via the query engine. The feedback loop throughout the system is continually active. Should D determine that categories are not representative of the searches requested, then A and B will suggest a re-analysis of the knowledge base in C. This is to ensure that the system is truly adaptive in a changing user environment. Without this flexibility, the tool would lay idle in preference to more customised alternatives.

The measure function, D, is the overall system controller. It provides the management interface to the operator, and the external feedback to the other modules. The feedback is based on analysis of the query engine usage logs. The logged parameters it monitors and acts on determine the system's ability to respond to changes in user behaviour. For instance, consider a user returning to the search page within a short time frame. This could be for many reasons, each of which would result in the communication of different messages. Firstly, the initial search could have given inaccurate search results. In this case user will probably refine their query, and D would indicate this to A, sending parameters to convey the specifics. A second reason for the subsequent visit is that the search proved very useful, provided a directly relevant result and the user simply wanted another query served. Two different results from seemingly the same user action, the second requiring no system update. D monitors usability, in terms of how often the service is used by users. Again there could be different reasons for the drop in usage,

someone could change projects (requiring less searching), they could be on leave, or in fact could be using other search tools. If it was only used to search certain subjects and generic search engines still used for others, then the category scope is not broad enough for the users interest and should be revised. D would indicate this requirement to A. In the future, D might iteroperate with system applications such as performance monitor tools to facilitate effective user modelling. Our early design, however, will encompass 'best guess' criteria based on past usage patterns.

The categoriser enables the cache based search system to be adaptive, in that metrics will report the success of user queries. Metrics will provide information as to the usefulness, usability and accuracy of the query engine by analysing usage patterns of the service. The system is fully automated and self modifying. User-input is not requested, the metrics are fully automated and the system adaptive to the results. This provides for consistency of analysis (human determined feedback is very subjective) and, with the right metrics, accurate user modelling. The combination of metadata and XML aware resources, with intelligent agents will give a tool capable of analysing and categorising both structured and unstructured data. Integrating system applications with intelligent agents can provide for a powerful metrics tool and the system will be capable of learning and modifying its actions.

The system described thus far is only what would be implemented on a single cache.

The most obvious additional requirement in a multinode implementation is to synchronise the actions of the autonomous adaptive nodes. In the context of our design this is expressed as the need to share knowledge of categories across all nodes participating in a particular query. There are several options for achieving this. The parent could simply send hourly cache digests [6] to its children, however the hashed information in current digests is insufficient for categorisation, and it is not yet clear how much more is required. Alternatively, it could periodically send its entire index, but this would be very bandwidth intensive (our current indexes are around 8% of the cache size). The best answer is probably to send the rules used by the categoriser, so that the children's categorisers can generate mappings between their own categories and those of their parent. A global categorisation scheme does not seem appropriate since this would not take advantage of user communities and localisation. It would also require universal agreement between cache administrators, which could impede the uptake of the proposal. Hence our design allows child nodes to define local categories, and map queries to its parents categories if the user indicates a need for broader search. A consequence of this decision is that the categorisation becomes weaker as the hierarchy is traversed. This is because the mappings will necessarily be less reliable than local categorisation based on full knowledge of the parent's index. The child would make requests of one or more parent categories in response to a user query of a local category. It can map the responses from its parent into its own categories before passing them back to the user but there may be responses missing which the local category would have caught but are not in the parent category (or the converse). Since many searches would be served by the local cache, and even a wider search would still serve more relevant results that generic search engines, this approach is considered to be well worth investigating.

The design of the implemented hierarchy might also include sibling searches prior to querying a parent, however, this would be just an ordering process and does not impact the need to provide category information to any cache which is forwarding queries.

It only remains to specify how the user indicates the desired breadth of search – at present we envisage a simple slider on the query engine interface page.

Our implementation

For an initial study we created a categorised index of a small 2-layer cache hierarchy used by our own research team. The cache hierarchy was implemented using MS Proxy Server and Microsoft Index Server was used to index the cache directories [7]. The index contained filename, metadata and unformatted text for each page, and was approx 8% of the total size of the cache. The indexer created an index for its local cache only. Queries could be made on the local cache, or on both the local cache and its parent. Queries of the parent cache were supported using a copy of the parent cache's index stored on the local cache, and updated on a daily basis. A standard Microsoft Index Server active server page query form was customised to display the subject areas that represented our categories. A smart spider [8] was used to categorise the cache indexes. As the categories were not adaptive it was not necessary to recategorise the index copied from the parent cache to match the local categories. A team of volunteers were requested to test the service. Users were requested to return feedback to test the implementation in terms of relevancy of results and their impression as to the idea of pre-searching categorisation. This generated useful qualitative feedback summarised in the next section. We also used the index server query logs to get an idea of how frequently users were entering the same query into different categories. Due to the small community involved the results are not

quantitatively useful, but they were sufficient to show that the query logs are a useful input for the category proposer in our design

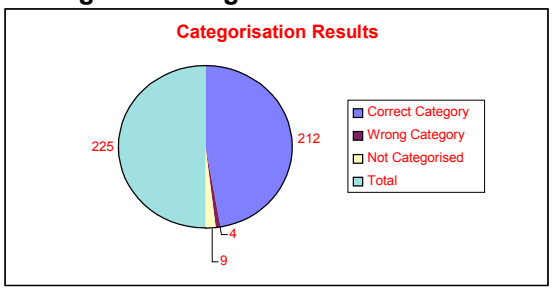
For a more controlled test, it was necessary to analyse the effectiveness of the spider on a known set of categories. The chosen source was an Encarta '98 CD [9]. The subject categories of the chosen pages were masked before the spider analysed the pages, and the results compared between the spider's categorisation and the encyclopaedia's classification. The source pages were chosen completely at random within a chosen classification on the CD, and in addition to this, extra pages were extracted from classifications not included in the spider's categories. This was to represent Web 'junk' pages, in that they weren't expected to fall into any of the chosen categories and hence searching via categories should not have resulted in one of the pages being served (thus testing whether irrelevant documents are presented to a user's query).

For each categorisation attempt we noted the number of words required in the spider vocabulary files, the time it took to analyse pages and the extent of human intervention required to help the spider, in time and activity. The categorisation was then characterised using several metrics. The metrics were the number of pages accepted into the correct category, the number of pages failed (not categorised that should have been), and those ambiguously categorised (accepted into more than one category, both where this double classification was valid and where one of the classifications was incorrect). Double classification is characteristic of subject classification [10], however, this is not necessarily a problem since many 'distinct' subject areas overlap. For instance, if the page subject is an ice-breaking trawler then from our categories, oceans and transport could both be deemed correct classifications. Sometimes, however, the second classification can be incorrect, made because a couple of words lead to a misleading diagnosis.

Results

The search engine was deemed easy to use and informative by the users who tested it. Perhaps the most important observation in the reports received was that all search results were deemed 100% relevant. Although the tests were carried out on a source which was not totally representative of Web contents (our cache matches our interests well), they do suggest the categorisation is of real benefit. The benefit would remain for searching a larger, less focused source. It would probably be greater. This is largely because the search topic is selected before a search actually takes place, hence users only search on a relevant information source, and only receive relevant results.

Figure 3 Categorisation Results



The key results from the Encarta based tests are shown in Figure 3. A 95% success rate in subject categorisation was achieved using the Smart Spider to analyse Encarta. There were 11 pages accepted into more than one category, and upon further investigation all categories were deemed valid. The spider used flat vocabulary files in order to associate words with categories and used rules files in order to specify acceptance criteria. Both rules files and vocabulary files could be extended, and by using them efficiently a 95% success rate was achieved with the spider's classifications. That is 95% of the source pages on the Encarta CD, under the headings we used, were found by the spider. The first time the spider categorised it's source led to a 70% success rate. Extending the vocabulary files and rule base, using a simple heuristic of one additional rule per new keyword, increased this number to first 80% and then 95%. Further extension using the simple rule generation strategy led to a reduction in performance because inappropriate pages were captured. The modification of the vocabulary files and the knowledge base was very time consuming. Testing required 14 hours to analyse the pages that were incorrectly classified and to make appropriate changes to the associated rule and vocabulary files. An agent with an automated learning capability would significantly reduce the operator effort required. However, by continually updating the words and associations within a particular subject, using an automateable rule generation heuristic, we can eventually lose accuracy. Spider administrators would need to intervene to perform more sophisticated rule generation if more accuracy was required, seeking the best trade-off between capturing more pages in a category and capturing irrelevant pages.

This is best discussed by example. Should 'ice-breaker trawler' be deemed to be correctly categorised as a subject within oceans, the categoriser could ensure that terms such as ice and ice breaker were within the oceans topic. However, this could result in information concerning the Antarctic and domestic ice-picks as being categorised within this topic. Correct inclusion could increase only slightly whilst incorrect inclusion would increase dramatically. Our aim was to find the trade-off between

source-page analysis accuracy and the strength of the subject categorisation.

We believe that the feedback that all search results were relevant to the search topic indicates the 95% accuracy reached was satisfactory.

The feedback and results received from our testing adds credibility to the concept that an efficient search engine fronting a scalable, categorised cache hierarchy would indeed be a tool worth developing.

Discussion

The work presented here has successfully shown that a managed cache directory provides an effective search base. In the future the directory will be exposed to a larger user community in order to obtain more meaningful web based measurements. In parallel with this we will implement the adaptation mechanism and investigate the sharing of categories between caches. We must also further automate the categoriser, increasing intelligence and reducing the human-hours required to 'teach' the agent. The end result should be an automated and truly distributed, scalable search engine that covers far more of the Web's cachable content, generates less traffic and is more attractive to end users, than current search tools.

In order for communications between caches to be effective, and to enable the largest traffic savings, cooperation between cache owners and implementers will be necessary. A standard is required in order for one cache to be able to understand and make use of the categories used by another. We have not yet progressed far enough to be able to identify the minimum information that must be shared, but it seems likely that caches will need to export their categorisation rules in an agreed format.

The process overheads of the proposed adaptive mechanism are not understood. Our experiments have run on old PCs (180 MHz Pentium), for up to 10 concurrent users, so we are confident the overheads are reasonable.

It is possible the categorisations will be unstable (oscillatory) for some user communities, so we may need to add an additional feedback loop to the design to damp oscillations. The identification of this type of instability could even be used to optimise the locality of user interests and further enhance response times.

Cache studies [11] have shown that there are different access patterns according to the culture of users, and the choice of server locations should be directly related to the findings of these studies. At present enough is known about the locality of user diversity to justify our choice of local categories rather than global categories, but further work is required to identify an optimum granularity, and this work is likely to be community specific. Highly diversified companies (conglomerates) will have many communities with only a small overlap of interests, whereas more focused companies are likely to have only one or a few distinct communities. The choice of location is thus company dependent.

Conclusions

We have proposed a design for a search tool based on adaptive categorisation of the contents of a cache hierarchy. We have demonstrated that automatic indexing and intelligent categorisation of Web cache content can provide an effective search tool. We have discussed the extra work required to fully justify our design, and the design issues that remain to be resolved. Initial indications are that our design is extremely promising and could potentially solve many current problems in web content location.

References

[1]http://www.computer.org/internet/v2n5/w5news-data.htm

[2]K. Bharat and A.Broder " A technique for measuring the relative size and overlap of public Web search engines". Computer Networks and ISDN Systems, 30, pp379. 1998

[3]C.M. Bowman, P.B. Danzig, D.R. Hardy, U. Manber, and M.F. Schwartz, The Harvest information discovery and access system, in: *Proc. of the 2nd World Wide Web Conference*

[4]http://www.missouri.edu/~libnh/ASI/tools.htm

[5]C Roadknight et al " Analysis of Artificial Neural Network Data Models" Proceedings of Intelligent Data Analysis '97.

[6]http://www.in.com.pl/linux/documentation/squid-1.2.beta22/FAQ/FAQ-16.html

[7]http://www.microsoft.com/products/prodref/590_ov.htm

[8]http://www.smart-spider.com/

[9]Microsoft Encarta '98 - British Edition

[10]N J Davies and M C Revett "Networked information management" BT Technology Journal 15 No2, April 1997, pp194

[11]"The influence of geographical and cultural issues on the cache proxy server workload" Almeida et al, Computer Networks and ISDN Systems, Vol. 30, issues 1-7. Article SP4.